



Desarrollo

```
https://github.com/SEBASBELMOS/Caoskol-Project
```



De momento se ha realizado **Docker Compose**, **Docker Swarm** y **Haproxy** está en proceso para **Docker Swarm**



El repositorio de GitHub esta en constante actualización, asegúrate de usar la última versión disponible - Asegúrate de hacer → `docker compose up --build -d` para **Docker Compose** y las **images** en **Docker Swarm**

▼ Directorio en VM

```
git clone https://github.com/SEBASBELMOS/Caoskol-Project.git
```

▼ VMs Azure (Opción si no se puede realizar en Vagrant)

```
sudo apt update  
sudo apt upgrade
```

```
sudo apt install apt-transport-https ca-certificates curl sof
```

```
curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sud
```

```
sudo add-apt-repository "deb [arch=amd64] https://download.do
```

```
sudo apt install docker-ce
```

```
sudo systemctl status docker
```

```
docker login
```

▼ Despliegue → Docker Compose

```
docker compose up -d
```

```
docker ps
```

```
root@servidorUbuntu:~/Caoskol-Project# docker compose up -d
WARN[0000] /root/Caoskol-Project/docker-compose.yml: 'version' is obsolete
[+] Running 4/4
✔ Container db                Started      0.0s
✔ Container notas             Running   0.0s
✔ Container usuarios           Running   0.0s
✔ Container web                Running   0.0s
root@servidorUbuntu:~/Caoskol-Project# docker ps
```

CONTAINER ID	IMAGE	NAMES	COMMAND	CREATED	STATUS	PORTS
3658f8ff99f0	caoskol-project-web	web	"docker-php-entrypoi..."	24 seconds ago	Up 23 seconds	0.0.0.0:8090->80/tcp, :::8090->80/tcp
087aee82cf9e	caoskol-project-usuarios	usuarios	"docker-entrypoint.s..."	24 seconds ago	Up 23 seconds	0.0.0.0:3030->3030/tcp, :::3030->3030/tcp
6386f7f83870	caoskol-project-notas	notas	"docker-entrypoint.s..."	24 seconds ago	Up 23 seconds	0.0.0.0:3031->3031/tcp, :::3031->3031/tcp
14e213f16cae	mysql:5.7	db	"docker-entrypoint.s..."	24 seconds ago	Up 4 seconds	33060/tcp, 0.0.0.0:32001->3306/tcp, :::32001->3306/tcp

▼ Prueba Funcionamiento



Acceder a <http://192.168.100.3:8090/> en un navegador (**Docker Compose**)



Acceder a <http://192.168.100.3:9080/> en un navegador (**Docker Swarm**)

▼ index.html ✓

East High

Resultados de Pruebas Anuales

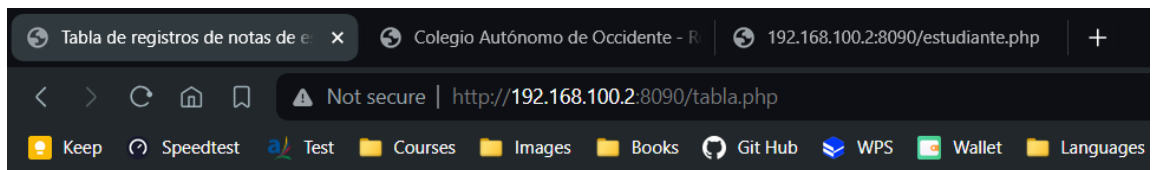
Iniciar Sesión

ID

Clave

Ingresar

▼ tabla.php ✓



Estadísticas notas de estudiantes

Estudiante	Grado	Nota de Matemáticas	Nota de Inglés	Nota de Ciencias	Número de Premios
Ana Quintero	10	85	78	92	10
MarÃa Fernanda Tello	11	90	82	88	11
Sebastian Belalcazar	10	90	90	80	9
Lewis Hamilton	9	88	85	91	10
Carlos Sainz	8	80	87	90	12

▼ estudiante.php ✓

Dashboard del Usuario

Not secure | http://192.168.100.2:8090/estudiante.php

Bienvenido

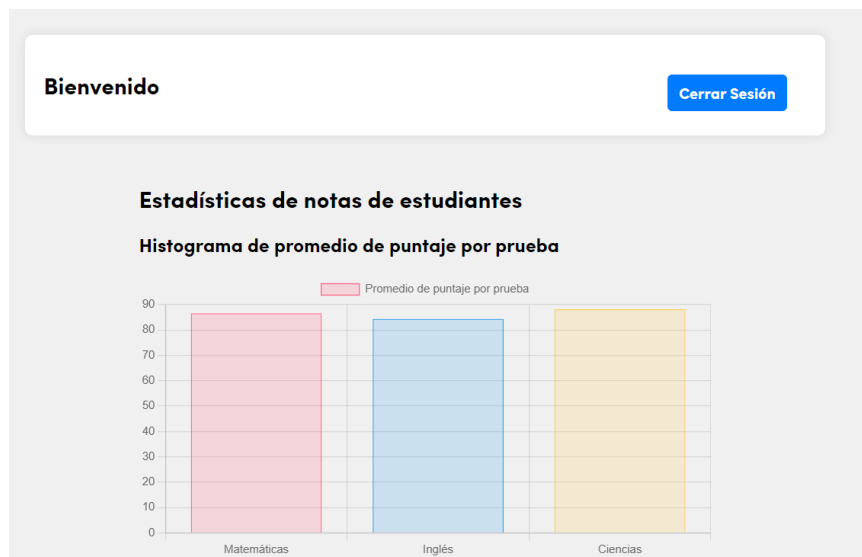
Cerrar Sesión

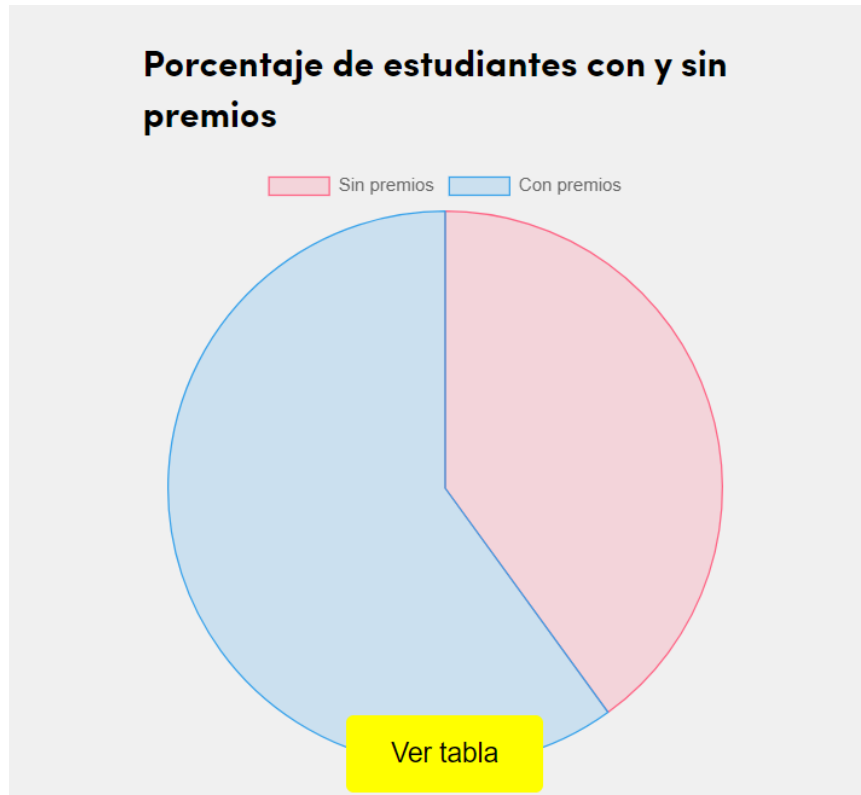
Sebastian Belalcazar

Grado: 10

Número de Premios	Puntaje de Matemáticas
0	90
Puntaje de Inglés	Puntaje de Ciencias
90	80

▼ profesor.php ✓





▼ Docker Swarm

▼ VM Principal (Cliente)

```
docker swarm leave --force
```

```
root@servidorUbuntu:~/Caoskol-Project# docker swarm leave --force
Node left the swarm.
```

```
docker swarm init --advertise-addr 192.168.100.3
```

```
root@servidorUbuntu:~/Caoskol-Project# docker swarm init --advertise-addr 192.168.100.2
Swarm initialized: current node (2du9jcnfjh68c8wpmipkgq90h) is now a manager.

To add a worker to this swarm, run the following command:

    docker swarm join --token SWMTKN-1-5jamldnuhx0qy19ctvtawjlcx0175elne9ocmx5x82k0zxxkoho-djwqn8lshohna9iz749flt0aa 192.168.100.2:2377

To add a manager to this swarm, run 'docker swarm join-token manager' and follow the instructions.
```

```
docker stack deploy -c docker-swarm.yml caoskol
```

```

root@servidorUbuntu:~/Caoskol-Project# docker stack deploy -c docker-swarm.yml caoskol
Since --detach=false was not specified, tasks will be created in the background.
In a future release, --detach=false will become the default.
Creating network caoskol_app-network
Creating service caoskol_notas
Creating service caoskol_web
Creating service caoskol_haproxy
Creating service caoskol_db
Creating service caoskol_usuarios

```

▼ VM Secundaria (Servidor)

```
docker swarm leave --force
```

```

vagrant@clienteUbuntu:~$ sudo I
root@clienteUbuntu:~# docker swarm leave --force
Node left the swarm.

```

```
docker swarm join --token SWMTKN-1-5jamldnuhx0qy19ctvtawj1
```

```

root@clienteUbuntu:~# docker swarm join --token SWMTKN-1-5jamldnuhx0qy19ctvtawj1cx0175elne9ocmx5x82k0zxkoho-djwqn8lshohn
a9iz749flt0aa 192.168.100.2:2377
This node joined a swarm as a worker.

```

▼ Images

```

docker build -t sebasbelmos/usuarios:latest ./appColegio/m
docker push sebasbelmos/usuarios:latest

```

```

docker build -t sebasbelmos/notas:latest ./appColegio/micr
docker push sebasbelmos/notas:latest

```

```













docker build -t sebasbelmos/web:latest ./APPWEB
docker push sebasbelmos/web:latest

```

```

docker build -t sebasbelmos/haproxy:latest ./haproxy
docker push sebasbelmos/haproxy:latest

```

 sebasbelmos/haproxy	latest	  Inactive	9 minutes ago	38.51 MB
 sebasbelmos/web	latest	  Inactive	21 minutes ago	177.19 MB
 sebasbelmos/notas	latest	  Inactive	21 minutes ago	406.3 MB
 sebasbelmos/usuarios	latest	  Inactive	21 minutes ago	406.3 MB

▼ Estado del Swarm desde el manager

```
docker node ls
```

```
root@servidorUbuntu:~/Caoskol-Project# docker node ls
ID                HOSTNAME        STATUS    AVAILABILITY    MANAGER STATUS    ENGINE VERSION
bovw8gpns9mab5uv3esft650v    clienteUbuntu    Ready     Active           Active             26.0.0
2du9jcnfjh68c8wpmipkgg90h *  servidorUbuntu  Ready     Active           Leader             26.0.0
root@servidorUbuntu:~/Caoskol-Project#
```

▼ Verificar los servicios desplegados

```
docker service ls
```

```
root@clienteUbuntu:~/Caoskol-Project# docker service ls
ID                NAME                MODE                REPLICAS    IMAGE                                  PORTS
4vwhwndhnude     caoskol_db          replicated          1/1         mysql:5.7                            *:32001->3306/tcp
s7z0rbm8oq9a     caoskol_haproxy     replicated          1/1         haproxy:latest                       *:6080->80/tcp
p87xdp2zaljb     caoskol_notas       replicated          2/2         sebasbelmos/notas:latest             *:3031->3031/tcp
omg1ckt66urc     caoskol_usuarios    replicated          2/2         sebasbelmos/usuarios:latest          *:3030->3030/tcp
54oxp4imi281     caoskol_web         replicated          3/3         sebasbelmos/web:latest               *:9080->80/tcp
root@clienteUbuntu:~/Caoskol-Project#
```

▼ Apagar el cluster

```
docker stack rm caoskol
```

▼ haproxy

```
docker config create haproxy_config haproxy.cfg
```

```
docker config rm haproxy_config
```

```
root@servidorUbuntu:~/Caoskol-Project/haproxy# docker config create haproxy_config_v1 haproxy.cfg
f4tt6ff0heb624nam4sdzytve
root@servidorUbuntu:~/Caoskol-Project/haproxy#
```

```
WARN NativeCodeLoader: Unable to load native-hadoop library for performance
WARN TaskSchedulerImpl: Initial job has not accepted any r
```

▼ Logs

```
docker service logs container_name
```

▼ JMeter

The screenshot shows the 'HTTP Request' configuration window in JMeter. The 'Name' field is set to 'HTTP Request'. The 'Basic' tab is selected, showing the 'Web Server' section with 'Protocol [http]:' set to 'http', 'Server Name or IP:' set to '192.168.100.3', and 'Port Number:' set to '9080'. The 'HTTP Request' section shows the method set to 'GET' and the 'Path:' field is empty. The 'Content encoding:' field is also empty. At the bottom, there are checkboxes for 'Redirect Automatically' (unchecked), 'Follow Redirects' (checked), 'Use KeepAlive' (checked), 'Use multipart/form-data' (unchecked), and 'Browser-compatible headers' (unchecked).

HTTP Request

The screenshot shows the 'View Results Tree' in JMeter. The left pane shows a list of 'HTTP Request' samples, all with green checkmarks. The right pane shows the details for the selected sample. The 'Sampler result' tab is active, displaying the following information:

- Thread Name: grupoPrueba 1-1
- Sample Start: 2024-05-23 18:46:45 COT
- Load time: 1
- Connect Time: 0
- Latency: 1
- Size in bytes: 3707
- Sent bytes: 123
- Headers size in bytes: 308
- Body size in bytes: 3399
- Sample Count: 1
- Error Count: 0
- Data type ("text"|"bin"|""): text
- Response code: 200
- Response message: OK

Below this, the 'HTTPSampleResult fields' are listed: 'ContentType: text/html' and 'DataEncoding: null'.

View Results Tree

The screenshot shows the 'Summary Report' window in JMeter. The 'Name' field is set to 'Summary Report'. The 'Log/Display Only' section has checkboxes for 'Errors' (unchecked) and 'Successes' (unchecked), and a 'Configure' button. Below this is a table with the following data:

Label ↓	# Samples	Average	Min	Max	Std. Dev.	Error %	Throughput	Received KB/sec	Sent KB/sec	Avg. Bytes
HTTP Request	10	6	1	48	13.81	0.00%	137.0/sec	495.92	16.45	3707.1
TOTAL	10	6	1	48	13.81	0.00%	137.0/sec	495.92	16.45	3707.1

Summary Report

Aggregate Report

Name:Aggregate Report

Comments:

Write results to file / Read from file

FilenameBrowse...Log/Display Only: ☐ Errors ☐ SuccessesConfigure

Label	# Samples	Average	Median	90% Line	95% Line	99% Line	Min	Maximum	Error %	Throughput	Received KB...	Sent KB/sec
HTTP Request	10	6	2	3	3	48	1	48	0.00%	137.0/sec	495.92	16.45
TOTAL	10	6	2	3	3	48	1	48	0.00%	137.0/sec	495.92	16.45

Aggregate Report



Tener en cuenta:

- El porcentaje de error disminuye entre más instancias se creen.
- A mas peticiones mas errores

▼ Escalabilidad

```
docker service scale caoskol_web=10
```

```
root@clienteUbuntu:~/Caoskol-Project# docker service scale caoskol_web=10
caoskol_web scaled to 10
overall progress: 10 out of 10 tasks
1/10: running [=====>]
2/10: running [=====>]
3/10: running [=====>]
4/10: running [=====>]
5/10: running [=====>]
6/10: running [=====>]
7/10: running [=====>]
8/10: running [=====>]
9/10: running [=====>]
10/10: running [=====>]
verify: Service caoskol_web converged
root@clienteUbuntu:~/Caoskol-Project#
```


```
docker service scale caoskol_web=1
```

```
root@clienteUbuntu:~/Caoskol-Project# docker service scale caoskol_web=1
caoskol_web scaled to 1
overall progress: 1 out of 1 tasks
1/1:
verify: Service caoskol_web converged
root@clienteUbuntu:~/Caoskol-Project#
```

▼ PySpark

▼ master

```
./start-master.sh --host 192.168.100.3 --port 7077 --webui
```

 **Spark Master at spark://192.168.100.3:7077**

URL: spark://192.168.100.3:7077
Alive Workers: 0
Cores in use: 0 Total, 0 Used
Memory in use: 0.0 B Total, 0.0 B Used
Resources in use:
Applications: 0 Running, 3 Completed
Drivers: 0 Running, 0 Completed
Status: ALIVE

▼ Workers (0)

Worker Id	Address	State	Cores	Memory	Resources
-----------	---------	-------	-------	--------	-----------

▼ Running Applications (0)

Application ID	Name	Cores	Memory per Executor	Resources Per Executor	Submitted Time	User	State	Duration
----------------	------	-------	---------------------	------------------------	----------------	------	-------	----------

▼ Completed Applications (3)

Application ID	Name	Cores	Memory per Executor	Resources Per Executor	Submitted Time	User	State	Duration
app-20240524030628-0002	ConsoleAnalysis	0	2.0 GiB		2024/05/24 03:06:28	root	FINISHED	24 s
app-20240524030529-0001	ConsoleAnalysis	0	1024.0 MiB		2024/05/24 03:05:29	root	FINISHED	38 s
app-20240524030412-0000	ConsoleAnalysis	0	1024.0 MiB		2024/05/24 03:04:12	root	FINISHED	2 s

```
./spark-submit --master spark://192.168.100.3:7077 --conf
```

▼ Pruebas con Pyspark

```
root@clienteUbuntu:~/SPARK/spark-3.5.1-bin-hadoop3/bin# ./
Python 3.10.12 (main, Nov 20 2023, 15:14:05) [GCC 11.4.0]
Type "help", "copyright", "credits" or "license" for more
24/05/24 02:29:39 WARN Utils: Your hostname, clienteUbuntu
24/05/24 02:29:39 WARN Utils: Set SPARK_LOCAL_IP if you ne
Setting default log level to "WARN".
To adjust logging level use sc.setLogLevel(newLevel). For
24/05/24 02:29:41 WARN NativeCodeLoader: Unable to load na
Welcome to
```

```
      _ _ _ _ _
     / _ _ \ _ _ _ _ _ / _ _ \
    _ \ _ \ _ _ \ _ _ \ / _ _ \ ' _ \
   / _ _ \ _ _ \ _ _ \ / _ _ \ _ _ \
    _ _ \
      version 3.5.1
```

```
Using Python version 3.10.12 (main, Nov 20 2023 15:14:05)
Spark context Web UI available at http://10.0.2.15:4040
Spark context available as 'sc' (master = local[*], app id = 
SparkSession available as 'spark'.
>>>
```

```
df = spark.read.options( header='True', inferSchema='True'
```

```

Welcome to
      /\_/\
     /__\/
Spark version 3.5.1

Using Python version 3.10.12 (main, Nov 20 2023 15:14:05)
Spark context Web UI available at http://10.0.2.15:4040
Spark context available as 'sc' (master = local[*], app id = local-1716517781965).
SparkSession available as 'spark'.
>>> df = spark.read.options( header='True', inferSchema='True' ).csv('/root/Caoskol-Project/US_Dataset.csv')
>>> df.count()
```

▼ `df.count()`

```
>>> df.count()
400
```

▼ `df.printSchema()`

```
>>> df.printSchema()
root
 |-- id: integer (nullable = true)
 |-- nombre: string (nullable = true)
 |-- grado: integer (nullable = true)
 |-- num_premios: integer (nullable = true)
 |-- programa: string (nullable = true)
 |-- puntaje_mat: integer (nullable = true)
 |-- puntaje_ing: integer (nullable = true)
 |-- puntaje_ciencias: integer (nullable = true)
```

▼ `df.select("grado").distinct().show()`

```
>>> df.select("grado").distinct().show()
+-----+
|grado|
+-----+
|    12|
|     6|
|     9|
|     8|
|     7|
|    10|
|    11|
+-----+
```

▼ Ejecutar `appPySpark.py`

```
./spark-submit /root/Caoskol-Project/appPySpark.py
```

```
root@clienteUbuntu:~/Caoskol-Project# ls
appColegio      db          LICENSE      'Proyecto - Modulo 1.pdf'  US_Dataset.xlsx
appPySpark.py  docker-compose.yml  node_modules  results.csv
APPWEB         docker-swarm.yml   package.json  US_Dataset.csv
dataset.py     haproxy          package-lock.json  US_Dataset_users.csv
root@clienteUbuntu:~/Caoskol-Project# cd results.csv/
root@clienteUbuntu:~/Caoskol-Project/results.csv# ls
part-000000-d1b3151b-bdbb-4f7e-9b9e-d4402c1e9776-c000.csv  _SUCCESS
```

? Como probamos que funciona ⬇

```
root@clienteUbuntu:~/Caoskol-Project# cd results.csv/
root@clienteUbuntu:~/Caoskol-Project/results.csv# ls
part-000000-d1b3151b-bdbb-4f7e-9b9e-d4402c1e9776-c000.csv  _SUCCESS
root@clienteUbuntu:~/Caoskol-Project/results.csv# cat part-000000-d1b3151b-bdbb-4f7e-9b9e-d4402c1e9776-c000.csv
grado,count
7,76
11,47
8,52
6,86
9,48
10,61
12,30
NULL,3
root@clienteUbuntu:~/Caoskol-Project/results.csv# |
```

▼ Ejecutar `analysis.py`

```
./spark-submit /root/Caoskol-Project/analysis.py
```

```
./spark-submit --master spark://192.168.100.3:7077 --conf
```

```

root@clienteUbuntu:~/Caoskol-Project# ls
analysis.py      clean_dataset_users.py  general_stats_results  package-lock.json      US_Dataset.xlsx
appColegio      correlacion_results     haproxy               promedio_por_grado_results
appPySpark.py   db                     LICENSE               'Proyecto - Modulo 1.pdf'
APPWEB          docker-compose.yml     node_modules          US_Dataset.csv
clean_dataset.py docker-swarm.yml        package.json          US_Dataset_users.csv
root@clienteUbuntu:~/Caoskol-Project#

```

como podemos ver en la carpeta del proyecto quedan nuevas carpetas en donde se pueden ver los resultados

```

+-----+-----+-----+-----+-----+-----+-----+-----+
|Promedio Matemáticas|Promedio Inglés|Promedio Ciencias|Promedio Premios|Desvío Estándar Matemáticas|Desvío Estándar Inglés|Desvío Estándar Ciencias|Desvío Estándar Premios|
+-----+-----+-----+-----+-----+-----+-----+-----+
|60.3775|60.735|NULL|1.5725|23.92752200968185|22.08776694511348|NULL|1.0829813983722113|
+-----+-----+-----+-----+-----+-----+-----+-----+
24/05/24 12:45:09 INFO FileSourceStrategy: Pushed Filters:

```

```

+-----+-----+-----+-----+-----+
|max_math|min_math|avg_math|max_awards|avg_awards|
+-----+-----+-----+-----+-----+
|100.0|20.0|60.3775|3.0|1.5725|
+-----+-----+-----+-----+-----+

```

```

+-----+-----+
|grado|count|
+-----+-----+
|8.0|52|
|7.0|76|
|11.0|47|
|10.0|61|
|6.0|86|
|9.0|48|
|12.0|30|
+-----+-----+

```

```
24/05/24 12:45:11 INFO CodeGenerator: Code generated in 4.546884 ms
+-----+-----+-----+-----+
|grado|Promedio Matemáticas|Promedio Inglés|Promedio Ciencias|
+-----+-----+-----+-----+
| 8.0|63.09615384615385|59.44230769230769|NULL|
| 7.0|57.85526315789474|60.671052631578945|NULL|
|11.0|57.97872340425532|65.57446808510639|NULL|
|10.0|58.78688524590164|61.114754098360656|NULL|
| 6.0|62.93023255813954|60.05813953488372|NULL|
| 9.0|62.166666666666664|58.291666666666664|NULL|
|12.0|58.866666666666667|60.63333333333333|NULL|
+-----+-----+-----+-----+
```

```
24/05/24 12:45:11 INFO TaskSchedulerImpl: Killing all running tasks in stage 16: Stage finished
24/05/24 12:45:11 INFO DAGScheduler: Job 11 finished: showString at NativeMethodAccessorImpl.java:6
24/05/24 12:45:11 INFO CodeGenerator: Code generated in 3.405307 ms
+-----+-----+-----+
|Correlación Premios-Matemáticas|Correlación Premios-Inglés|Correlación Premios-Ciencias|
+-----+-----+-----+
| -0.02141959461403...| -0.04184099492909237|NULL|
+-----+-----+-----+

24/05/24 12:45:12 INFO FileSourceStrategy: Pushed Filters:
24/05/24 12:45:12 INFO FileSourceStrategy: Post-Scan Filters:
24/05/24 12:45:12 INFO MemoryStore: Block broadcast_19 stored as values in memory (estimated size 1
```

▼ Análisis

1. **Selección del dataset objetivo:** se debe seleccionar el dataset a usar y describirlo.

Decidimos tomar como dataset la base de datos de los estudiantes del Colegio Autónomo de Occidente, donde nos muestran las notas y los premios obtenidos por estos estudiantes.

Es un dataset con notas de alrededor de 400 estudiantes, donde se encuentran los puntajes en pruebas de matemáticas, inglés y ciencias. Todos los estudiantes de bachillerato realizan unas pruebas anuales de competencias. Como preparación para estos exámenes, los estudiantes pueden escribir en un tipo de programa de estudio diferente al general; esta información debe estar almacenada, pero no visible (Por políticas de seguridad). También, los estudiantes que hayan cumplido con ciertos criterios de excelencia en el año lectivo, pueden ganar uno o más premios.

Seleccionar el dataset del Colegio Autónomo de Occidente es valioso porque proporciona una vista integral del rendimiento académico en matemáticas, inglés y ciencias de alrededor de 400 estudiantes, permitiendo análisis estadísticos. La inclusión de información sobre premios y programas de

estudio específicos, aunque no visibles, permite evaluar la influencia de estos factores en el rendimiento y el reconocimiento por excelencia. Este análisis puede identificar factores de éxito y guiar mejoras en políticas educativas y estrategias pedagógicas. Además, es relevante para la investigación educativa y permite construir modelos predictivos que anticipen el rendimiento y necesidades de apoyo de los estudiantes, todo esto asegurando la confidencialidad de los datos sensibles.

2. **Generación y selección de alternativas de solución:** se deben evaluar diferentes alternativas de solución para el empaquetado de la aplicación en contenedores y el despliegue en un cluster de procesamiento de datos distribuido. Esto implica evaluar diferentes tecnologías y herramientas disponibles, y seleccionar las que mejor se adapten a las necesidades del proyecto.

Decidimos utilizar la plataforma de procesamiento de datos distribuidos Apache Spark en este proyecto debido a su rendimiento superior, escalabilidad, flexibilidad, facilidad de uso, y capacidades avanzadas de procesamiento de datos en memoria y en tiempo real. Además nos permite manejar eficientemente grandes volúmenes de datos y realizar análisis complejos de manera efectiva.

- La aplicación debe poder generar reportes y visualizarlos en un dashboard.

PySpark es la API de Python para Apache Spark y puede ser utilizado para procesar grandes volúmenes de datos y generar reportes que luego pueden ser visualizados en un dashboard.

Para crear el dashboard, podemos utilizar herramientas de visualización como Dash de Plotly, Flask, o Streamlit.

En nuestro caso usaremos Flask.

3. **Definición de la arquitectura completa del sistema:** se debe definir una arquitectura que permita el empaquetado de una aplicación basada en microservicios y API Rest en un cluster de contenedores y el despliegue de

la aplicación de análisis de datos en un cluster de procesamiento de datos distribuido. Esto implica definir los componentes necesarios, los servicios que se deben implementar, y las tecnologías a utilizar.

Componentes Principales:

Microservicios y API Rest:

- **Docker:** Para contenerizar los microservicios.
- **Frameworks:** Flask o FastAPI para construir APIs Rest.
- **Base de Datos:** SQL.

Aplicación de Análisis de Datos:

- **Apache Spark:** Para procesamiento de datos distribuidos.
- **PySpark:** Interfaz de programación en Python.

Clúster de Procesamiento de Datos Distribuido:

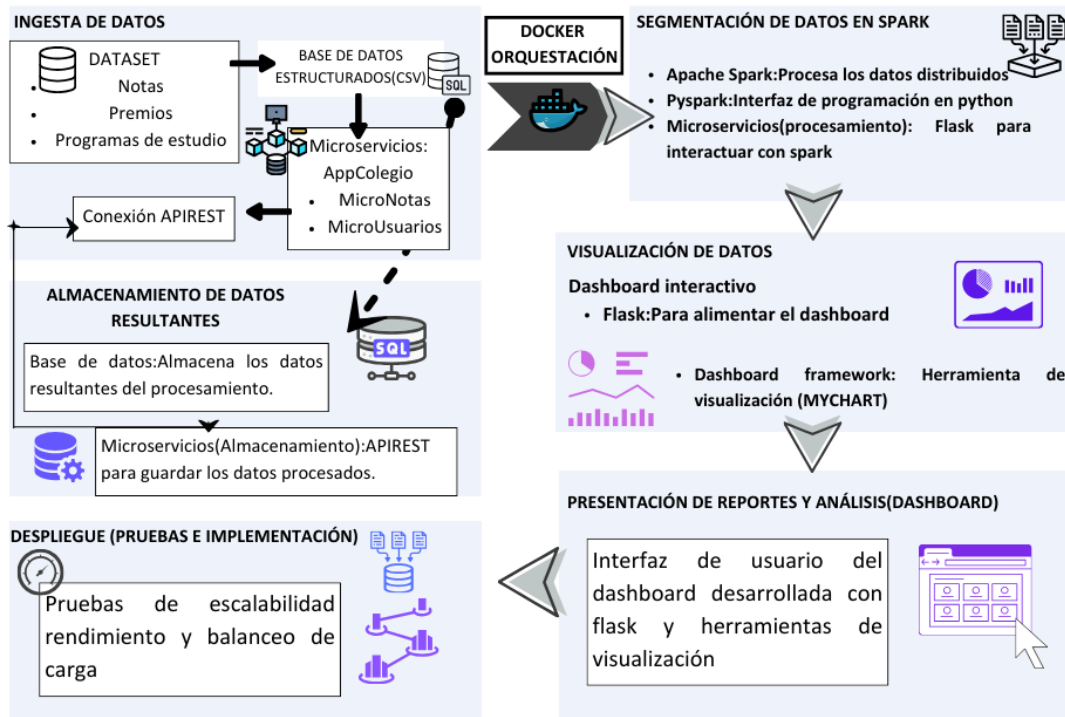
- **Hadoop YARN:** Gestión de recursos y ejecución.
- **Spark on YARN:** Ejecución de trabajos de Spark.

Integración y Comunicación:

- **CI/CD Pipeline:** Integración y despliegue continuo.

▼ DESARROLLO DE DISEÑO

Diagrama de los componentes a utilizar(Pipeline)



1. Componentes y Servicios a Implementar(PIPELINE)

Para el proyecto de análisis de datos de los estudiantes del Colegio Autónomo de Occidente, se utilizarán los siguientes componentes y servicios:

1. Contenedores y Orquestación:

- **Docker:** Para contener los microservicios y asegurar que cada componente sea independiente y fácilmente desplegable.

2. Microservicios y API Rest:

- **Flask:** Para desarrollar APIs RESTful que manejarán las solicitudes y respuestas entre los diferentes servicios.
- **FastAPI:** Alternativa a Flask para APIs de alto rendimiento.

3. Base de Datos:

- **Base de datos SQL :** para almacenar datos estructurados sobre estudiantes, notas y premios.

4. **Procesamiento de Datos Distribuido:**

- **Apache Spark:** Para el procesamiento de datos en un entorno distribuido.
- **PySpark:** API de Python para interactuar con Apache Spark y realizar análisis de datos.

5. **Visualización de Datos:**

- **Flask:** Para servir el dashboard de visualización.
- **Jupyter Notebooks:** Para desarrollo y pruebas interactivas.

6. **CI/CD Pipeline:**

- **GitLab :** Para la integración y el despliegue continuo, asegurando que los cambios en el código se prueben y desplieguen automáticamente.

2. 0 Relación y Flujo de Trabajo entre los Componentes

1. **Datos de Entrada:**

- Los datos de los estudiantes (notas, premios, programas de estudio) son almacenados en la base de datos.

2. **Extracción y Procesamiento de Datos:**

- Un microservicio desarrollado con Flask extrae los datos relevantes de la base de datos.
- Los datos extraídos son enviados a Apache Spark para procesamiento mediante PySpark.

3. **Análisis y Generación de Reportes:**

- Apache Spark procesa los datos, realiza análisis complejos y genera resultados.
- Los resultados del análisis son enviados a otro microservicio que formatea estos datos para visualización.

4. **Visualización de Datos:**

- Los datos procesados se visualizan en un dashboard interactivo creado desde el código de programación

- Los usuarios pueden interactuar con el dashboard para ver reportes detallados.

5. Despliegue y Orquestación:

- Todos los microservicios y componentes son contenerizados usando Docker.

6. CI/CD Pipeline:

- GitLab automatizan la integración y el despliegue, probando cada cambio en el código y desplegando los servicios.

3. Descripción de los Componentes

1. Docker:

- **Función:** Contener las aplicaciones para asegurar independencia y facilidad de despliegue.
- **Datos:** Código de la aplicación, dependencias, configuración.

1. Flask/FastAPI:

- **Función:** Desarrollo de APIs RESTful para comunicación entre servicios.
- **Datos:** Solicitudes HTTP, respuestas con datos procesados.

2. SQL:

- **Función:** Almacenamiento de datos estructurados.
- **Datos:** Información de estudiantes, notas, premios, programas de estudio.

3. Apache Spark/PySpark:

- **Función:** Procesamiento distribuido de datos y análisis complejo.
- **Datos:** Datos de entrada desde PostgreSQL, resultados de análisis.

4. GitLab :

- **Función:** Integración y despliegue continuo.
- **Datos:** Código fuente, scripts de despliegue, registros de pruebas y despliegue.

Esta estructura asegura un flujo de trabajo eficiente y escalable para el análisis de datos, desde la extracción hasta la visualización, garantizando que cada componente funcione de manera independiente

Diagrama de despliegue: se debe crear un diagrama que muestre cómo se despliegan los diferentes componentes en el sistema, incluyendo los nodos en los que se ejecutan.

Diagrama de Despliegue

El diagrama de despliegue muestra cómo se distribuyen los componentes del sistema en los nodos de hardware y los contenedores que se ejecutan en cada uno.

Descripción del Despliegue

Servidor Ubuntu:

Contenedores desplegados:

db: Contenedor de la base de datos.

notas: Contenedor del microservicio de notas.

usuarios: Contenedor del microservicio de usuarios.

web: Contenedor del servicio web o dashboard.

Detalles de Contenedores

db: Base de datos SQL que almacena la información de estudiantes, notas y premios.

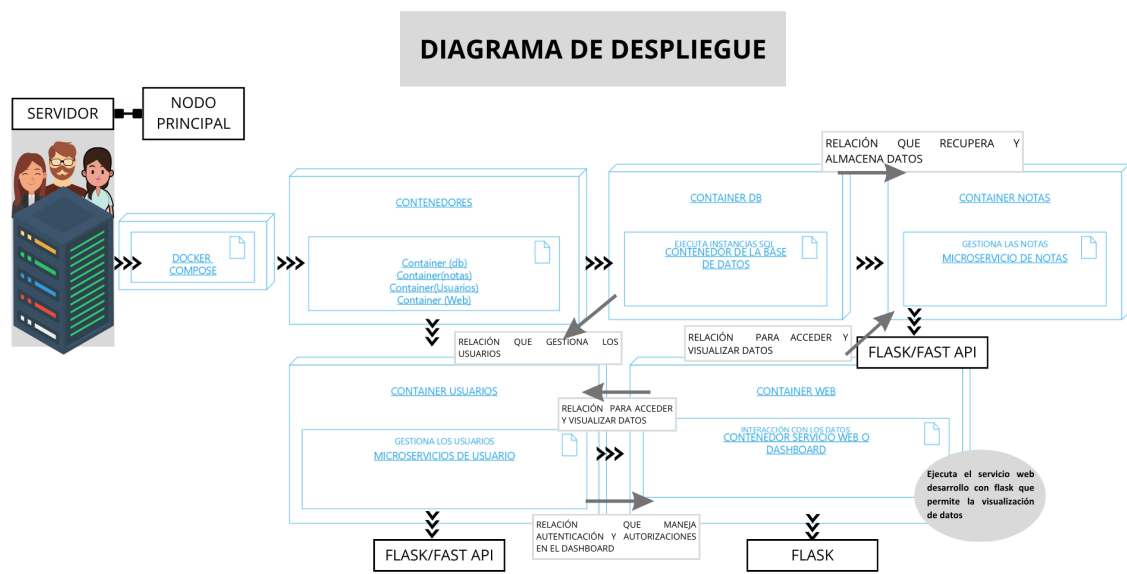
notas: Microservicio que maneja la lógica relacionada con las notas de los estudiantes.

usuarios: Microservicio que maneja la autenticación y gestión de usuarios.

web: Servicio web que sirve el dashboard y permite la interacción con los datos procesados.

Diagrama de Despliegue

El diagrama de despliegue puede ser representado de la siguiente manera:



Explicación del Diagrama

Servidor Ubuntu: Este es el nodo principal donde se ejecuta el Docker Engine.

Docker Engine: Motor de contenedores que gestiona la ejecución de contenedores.

Contenedores:

db: Ejecuta una instancia de PostgreSQL para la base de datos.

notas: Ejecuta un microservicio desarrollado con Flask/FastAPI para la gestión de notas.

usuarios: Ejecuta un microservicio desarrollado con Flask/FastAPI para la gestión de usuarios.

web: Ejecuta el servicio web desarrollado con Flask que sirve el dashboard para la visualización de datos.

Conexiones entre Contenedores

db:

Conectado a notas y usuarios para almacenar y recuperar datos.
notas:

Conectado a db para recuperar y almacenar información de notas.

Conectado a web para servir datos procesados al dashboard.

usuarios:

Conectado a db para autenticación y gestión de usuarios.

Conectado a web para manejar autenticación y autorizaciones en el dashboard.

web:

Conectado a notas y usuarios para acceder a los datos necesarios para la visualización y gestión del dashboard.